Amendments to the claims

This listing of claims will replace all prior versions, and listings, of claims in the application :

1. (presently amended) A method for retrieving and modifying data elements on a shared medium, comprising:

receiving at a shared medium controller, and from multiple clients, ~~serialized~~ transaction requests, each of said transaction requests being for at least one of reading and writing a data element on said shared medium;

adding said transaction requests to an execution queue in order;

retrieving from said execution queue all read transaction requests for said data element;

executing all of said read transaction requests for said data element in parallel, until a write transaction request for said data element is retrieved from said execution queue, and according to said order;

executing said write transaction request for said data element and queuing all incoming read transaction requests for said data element until said write transaction request is completed;

determining and changing a state associated with said data element for each of said executing of said transaction requests, said state stored in said shared medium controller; and

whereby data retrieval transactions on said data element, originating from one or more clients, are not compromised by data update transactions on said data element originating from other clients.

2. (previously amended) The method as claimed in claim 1, further comprising:

checking whether a version state associated with said data element is locked;

if said version state is locked,

adding said transaction request for said data element to an execution queue;

if said version state is not locked,

executing said transaction request right away.

3. (original) The method as claimed in claim 1, wherein said data elements are referred to as pointers defining an address range.

4. (original) The method as claimed in claim 3, wherein a read transaction for multiple data elements is executed for all of said data elements which are not waiting for a pending write transaction, and following completion of said pending write transaction, remaining data elements are read.

5. (previously amended) The method as claimed in claim 1, wherein said read transaction request and execution comprises:

providing to said shared medium controller a client-stored version state information associated with said data element to be read;

comparing said client-stored version state information associated with said data element with a shared medium-stored state information;

sending back to originating client a confirmation that said client-stored version state information and the contents of said data element are accurate.

6. (original) The method as claimed in claim 1, wherein said shared medium controller is physically located with the shared medium.

7. (previously amended) The method as claimed in claim 2, wherein said version state is locked following a transaction request originating from a client.

8. (previously amended) The method as claimed in claim 3, wherein said shared medium controller maintains a list of version state information associated with said data elements on said shared medium and following modification of said data elements, said

shared medium controller creates a new data structure in said list containing a new version state of said data elements and associating said new version state with a part of said data elements that has been modified.

9. (original) The method as claimed in claim 8, wherein if said data elements being modified are associated with multiple separate data structures containing version states, creating a new single data structure in said list associated with all modified said data elements and removing said multiple separate data structures from said list.

10. (previously amended) The method as claimed in claim 8, wherein said version state is an initial version number and wherein said initial version number is incremented to obtain said new version state.

11. (original) The method as claimed in claim 8, wherein said list of data structures is a double linked binary tree list.

12. (presently amended) A method for executing a common task in a clustered computing environment comprising a plurality of computers interconnected to collaborate on said common task, said plurality of computers including at least a client computer and a shared storage medium storing data elements, comprising:

receiving at a shared medium controller, and from said plurality of client computers working on a same task, ~~serialized~~ transaction requests, each of said transaction requests being for at least one of reading and writing a data element on said shared storage medium;

adding said transaction requests to an execution queue in order;

retrieving from said execution queue all read transaction requests for said data element;

executing all said read transaction requests for said data element in parallel, until a write transaction request for said data element is retrieved from said execution queue, and according to said order;

executing said write transaction request for said data element and queuing all incoming read transaction requests for said data element in said execution queue in order until said write transaction request is completed;

determining and changing a state associated with said data element for each of said executing of said transaction requests, said state stored in said shared medium controller; and

at least one of said plurality of computers modifying said data element stored on said shared storage medium;

said client computer retrieving said data element stored and using said data element stored to execute said common task;

whereby data retrieval transactions on said data element, originating from one or more client computers, are not compromised by data update transactions on said data element originating from other client computers.

13. (previously amended) The method as claimed in claim 12, further comprising:

checking whether a version state associated with said data element is locked;

if said version state is locked,

adding said transaction request for said data element to an execution queue;

if said version state is not locked,

executing said transaction request right away.

14. (previously presented) The method as claimed in claim 12, wherein said data elements are referred to as pointers defining an address range.

15. (previously presented) The method as claimed in claim 14, wherein a read transaction for multiple data elements is executed for all of said data elements which are not waiting for a pending write transaction, and following completion of said pending write transaction, remaining data elements are read.

16. (previously amended) The method as claimed in claim 12, wherein said read transaction request and execution comprises:

providing to said shared medium controller a client-stored version state information associated with said data element to be read;

comparing said client-stored version state information associated with said data element with a shared medium-stored state information;

sending back to originating client a confirmation that said client-stored version state information and the contents of said data element are accurate.

17. (previously presented) The method as claimed in claim 12, wherein said shared medium controller is physically located with the shared medium.

18. (previously amended) The method as claimed in claim 13, wherein said version state is locked following a transaction request originating from a client.

19. (previously amended) The method as claimed in claim 14, wherein said shared medium controller maintains a list of version state information associated with said data elements on said shared medium and following modification of said data elements, said shared medium controller creates a new data structure in said list containing a new version state of said data elements and associating said new version state with a part of said data elements that has been modified.

20. (previously presented) The method as claimed in claim 19, wherein if said data elements being modified are associated with multiple separate data structures containing version states, creating a new single data structure in said list associated with all modified said data elements and removing said multiple separate data structures from said list.

21. (previously amended) The method as claimed in claim 19, wherein said version state is an initial version number and wherein said initial version number is incremented to obtain said new version state.

22. (previously presented) The method as claimed in claim 19, wherein said list of data structures is a double linked binary tree list.